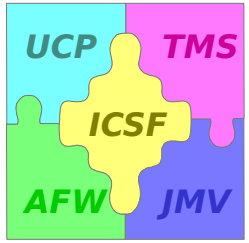
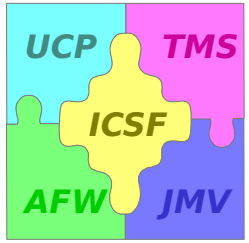


Universal Communications Processor (UCP)

Anthony Alston

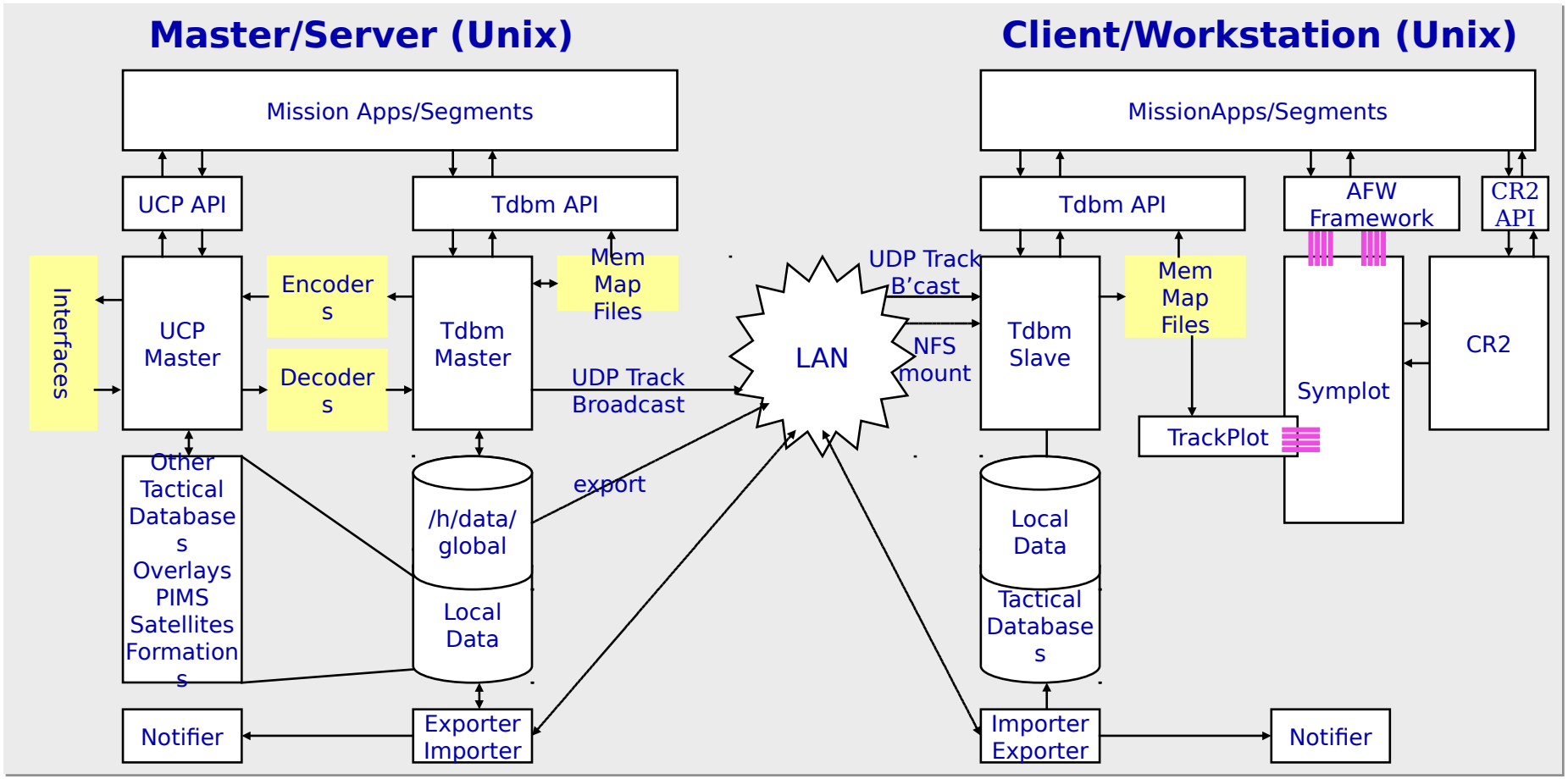
LOGICON INRI

UCP Agenda

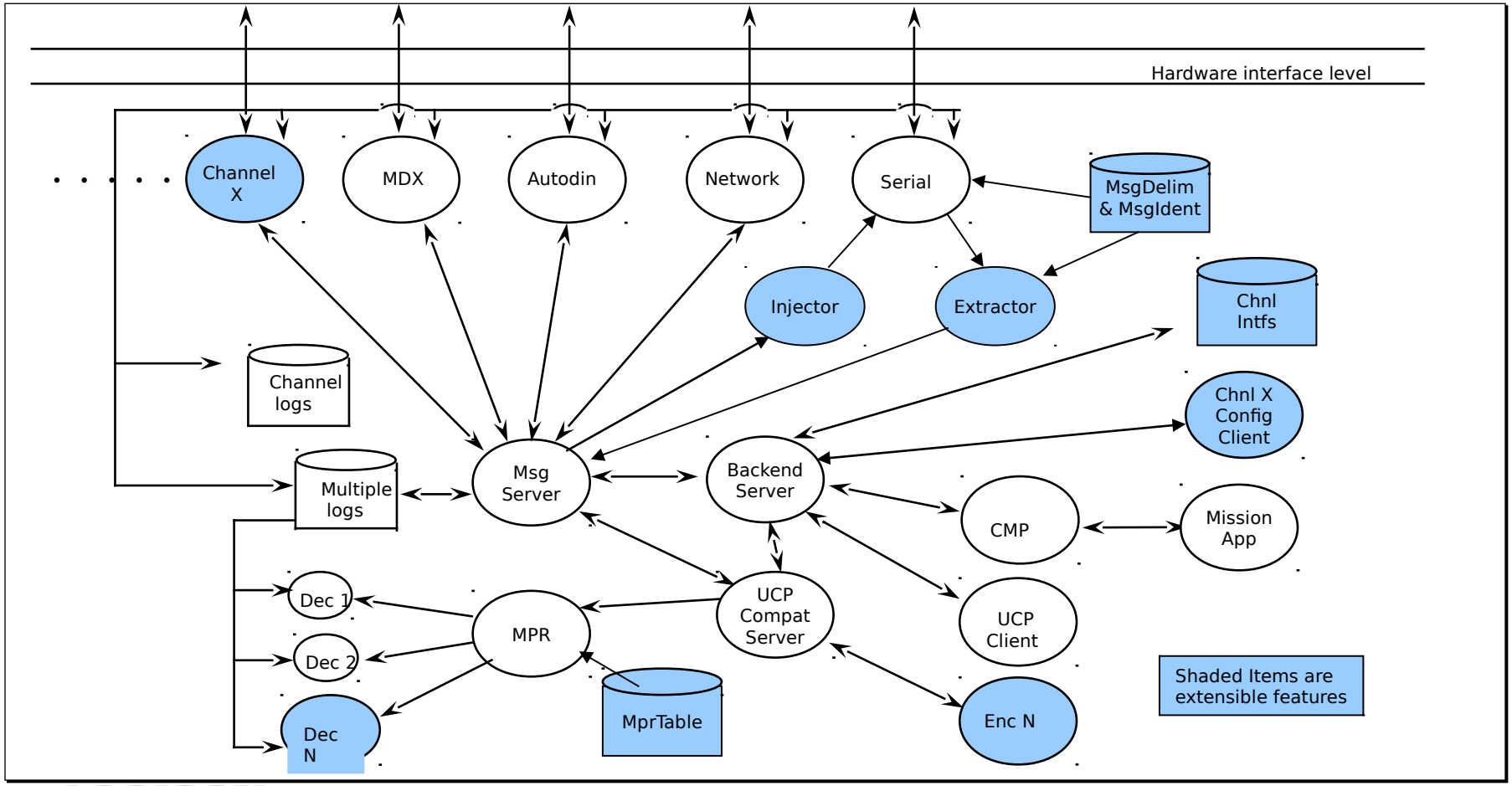


- *Overview*
- *Architectures*
- *Features*
- *Using UCP in Mission Applications*
  - *C API example*
  - *Java API example*

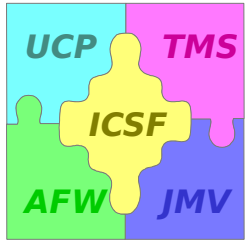
ICSF 4.X Architecture



UCP 4.X

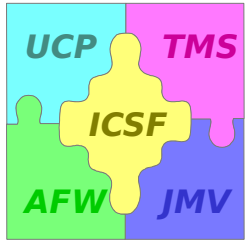

Hardware interface level
Channel X
MDX
Autodin
Network
Serial
MsgDelim & MsgIdent
Injector
Extractor
Chnl Intfs
Channel logs
Chnl X Config Client
Multiple logs
Msg Server
Backend Server
CMP
Mission App
Dec 1
Dec 2
Dec N
MPR
UCP Compat Server
UCP Client
MprTable
Enc N
Shaded Items are extensible features

4.X Upgrades



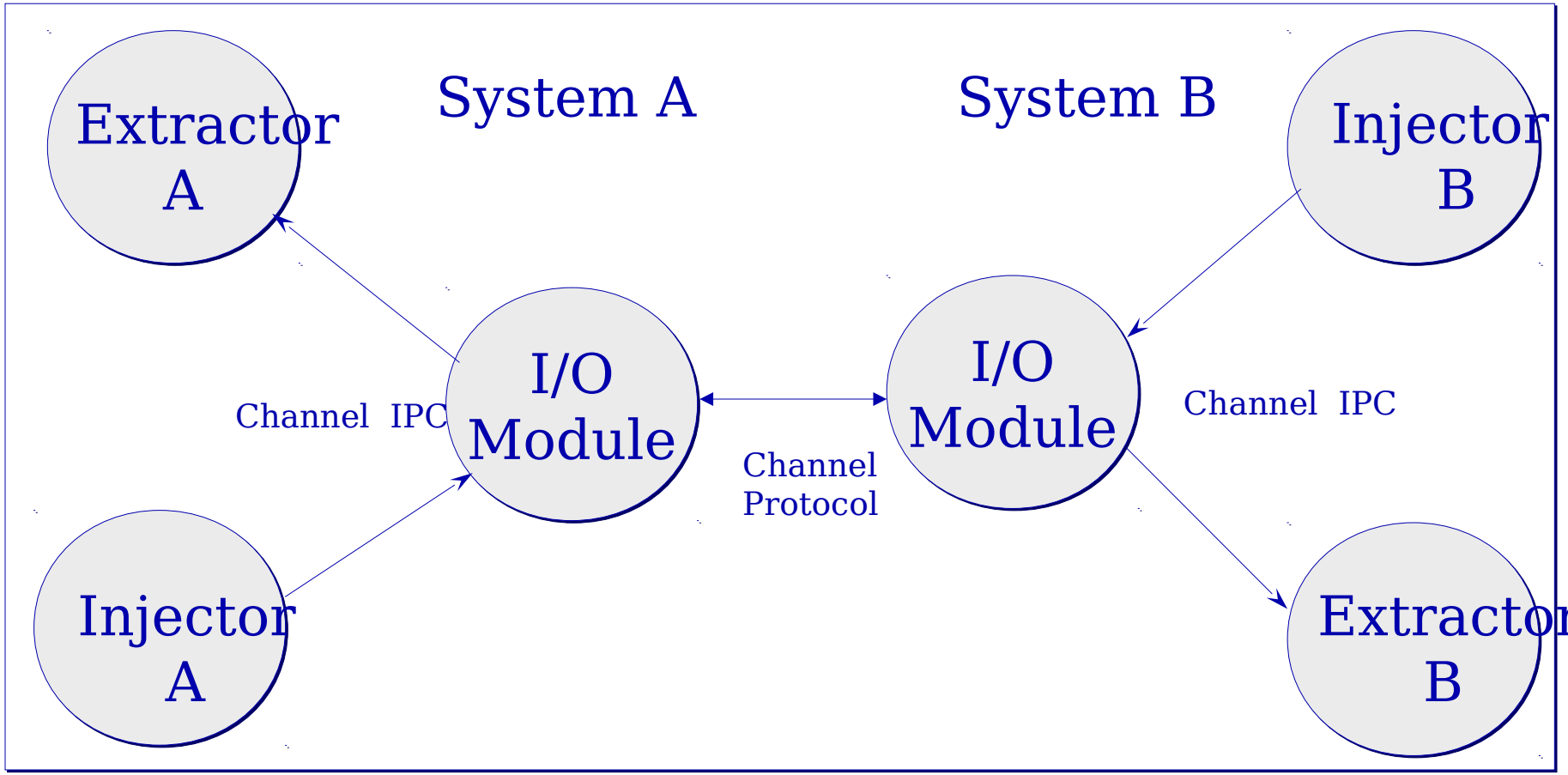
- *Version Translation*
- *NT Client Support (Java Bindings, Java GUIs)*
- *Java 1.2*

UCP 4.X Extensible Features



- *Adding Comms Channels*
- *Adding Comms Channels Injectors and Extractors*
- *Adding Message Identification Rules*
- *Adding Message Delimiting Rules*
- *Adding Message Processing Encoders and Decoders*

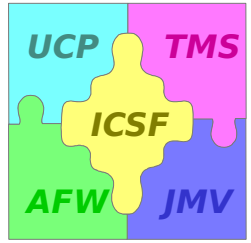
Channel Architecture



I/O Modules, Injectors & Extractors

- *I/O Module implements a particular comms protocol (e.g. IDS 8648, WS19702, etc.)*
- *Receive*
  - *I/O Module receives raw data from devices via the comms protocol and delimits messages*
  - *I/O Module passes messages to an extractor process via a defined Channel Interprocess Communication (Channel IPC).*
- *Transmit*
  - *Injector process passes a message to the I/O Module via a defined Channel IPC.*
  - *I/O Module takes the messages and transmits the data on the device via the comms protocol.*

I/O Modules, Injectors & Extractors



I/O Modules

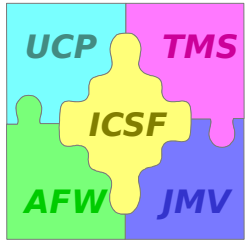
GenDuplex (Serial)
Mdx
Network
RFNet
etc.

Injectors & Extractors

Formatted Msgs
CSI
COP
Tracks
etc.

I/O Modules can be paired with Injectors & Extractors at runtime.

UCP Channel Tool

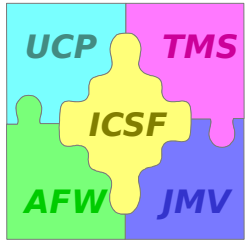


- *UCPChnlTool [-add, -del] [channel, injector, extractor] -file chnl_file : for adding/deleting a channel, an injector, or an extractor to/from the system.*
  - *Invoked from a Segments PostInstall (or the command line)*
  - *chnl_file is in resource file format (label:value)*
  - *Tool handle conflicts, database full, etc.*

UCP Message Tool

- *UCPMsgTool [-add, -del] [delim, ident, decoder] -file msg_file - for adding/deleting a message delimiters, identifiers and decoders to/from the system.*
  - *Invoked from a Segments PostInstall (or the command line)*
  - *msg_file is in resource file format (label:value)*
  - *Tool handles conflicts, database full, etc.*

UCP Plug In Example

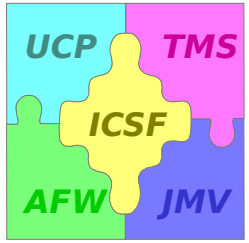


- Channel
  - UCPChannelConfigFile
- Message Type
  - UCPMsgtype-GOLDRPT
  - UCPMsgType-TACELNT
- Message Decoder
  - UCPMsgDecodersConfigFile

UCP API Comparison

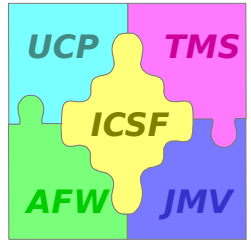
- 3.1 - C
  - MLOG mlog;
  - mlog.dtg = time(0);
- 3.3 - C
  - UCP_MSG_WRAP msg_wrap = UCPMsgWrapCreate();
  - UCPMsgWrapSetDtg(msg_wrap,time(0));
- 4.X - Java
  - IUCPMessage msg = UCPMsgFactory.newUcpMsg();
  - msg.setMsgDtg(time(0));

Using UCP API Services



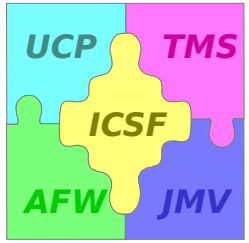
- *Composing a Message*
  - *UCPMsgCreate - creates a empty message object.*
  - *UCPMsgSubmit - submits a received message for further processing/decoding.*
  - *UCPMsgRelease - release a message via a channel.*
- *Channel updates*
  - *UCPChnlLoad - load the current list of channels*
  - *UCPChnlUpdate - submits an updated channel to the server.*
- *Context*
  - *UCPCMsgTypeLoad - load the current list of supported message types.*
  - *UCPCGetSvrLanHost - list of authorized UCP server/client machines.*

Using UCP API Services



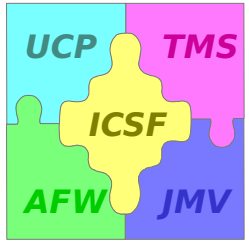
- *UCP Event Notification*
  - *UCPCGetNextEvent - Clients can register for the following events.*
    - *Channel Updates*
    - *Message Receipt*
    - *Message Transmission*
    - *Server Status Changes*
    - *Server Data Updates*

UCP Sample Code - C (1)




```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <malloc.h>
#include <UCP/UCPC.h>           /* UCPC Public API */
#include <UCP/UCPMsg.h>         /* UCPMsg Public API */
#include <UCP/UCPMsgType.h>     /* UCPMsgType Public API */

int
main(int argc,char *argv[])
{
    UCPC ucpc;
    UCP_EVENT ucp_event;
    UCP_MSG msgobj;
    UCP_MSG_TYPE msg_type;
    UCP_MSG_TYPE_LIST msg_type_list;
    char **msgobj_body;
    int event_type, num_pending, lines;
```

UCP Sample Code - C (2)

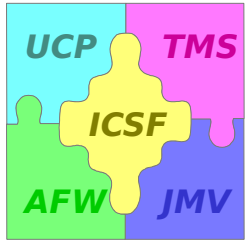



```
/* Attach to the UCP server and register the Event. */
ucpc = UCPCAttach();

/* Test server is running ok. */
if(UCP_SERVER_DOWN==UCPCServerStatus(ucpc)) {
    printf("UCPCserverStatus() is UCP_SERVER_DOWN, exiting.\n");
    return(0);
}
/* Create Event and fill in the attributes. */
ucp_event = UCPEventCreate();
msg_type_list = UCPCMsgTypesLoad(ucpc);

/* Registering for all message types.*/
for (msg_type  = UCPMsgTypeListGetFirstMsgType(msg_type_list);
     msg_type != NULL;
     msg_type  = UCPMsgTypeListGetNextMsgType(msg_type_list)) {
    UCPEventSetMsgType(ucp_event, UCPMsgTypeGetString(msg_type));
}
UCPMsgTypeListDestroy(msg_type_list);
```

UCP Sample Code - C (3)

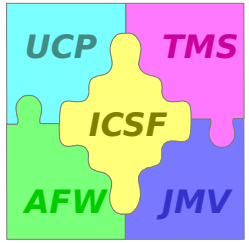
```
/* Set event flags for retrieving pending message and server/channel events. */
UCPEventSetPending(ucp_event, 0); /* 1=True, 0=False */
UCPEventSetServerChangeStatus(ucp_event, 1); /* 1=True, 0=False */
UCPEventSetChannelChangeStatus(ucp_event, 1); /* 1=True, 0=False */
num_pending = UCPCRegisterEvent(ucpc, ucp_event);
UCPEventDestroy(ucp_event);

/* Wait for new events */
while (ucp_event = UCPCGetNextEvent(ucpc)) {
    UCPEventGetAttrib(ucp_event, UCP_EVENT_TYPE, &event_type, NULL);
    switch (event_type) {
    case UCP_MESSAGE_EVENT:
        UCPEventGetAttrib(ucp_event, UCP_EVENT_MSG, &msgobj, NULL);
        if (msgobj != NULL) {
            UCPMsgGetAttrib(msgobj, UCP_MSG_RAW, &msgobj_body, &lines, NULL);
            for (int i=0;i<lines;i++) {
                printf("%s\n",msgobj_body[i]);
                free(msgobj_body[i]);
            }
            free(msgobj_body);
            UCPMsgDestroy(msgobj);
        }
        break;
```

UCP Sample Code - C (4)



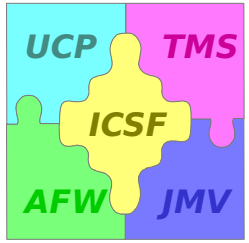

```
        case UCP_SERVER_CHANGE_EVENT:
            printf("Event type is UCP_SERVER_CHANGE_EVENT.\n");
            break;
        case UCP_CHANNEL_CHANGE_EVENT:
            printf("Event type is UCP_CHANNEL_CHANGE_EVENT.\n");
            break;
        case UCP_SERVER_DOWN_EVENT:
            printf("Event type is UCP_SERVER_DOWN_EVENT.\n");
            UCPCDetach(ucpc);
            return(0);
        case UCP_UNKNOWN_EVENT:
            printf("Event type is UCP_UNKNOWN_EVENT.\n");
            break;
        }
        UCPEventDestroy(ucp_event);
    }
    UCPCDetach(ucpc);
    return(0);
}
```

UCP Sample Code – C Makefile



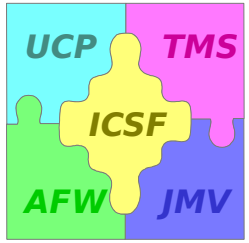

```
all : progs

progs : UCPEventApiTest.exe

UCPEventApiTest.exe : UCPEventApiTest.o
   link /LIBPATH:$(MSDEV_LIBS) -out:UCPEventApiTest.exe
   UCPEventApiTest.o $(UCPDV_LIB_HOME)\UCP.lib


    del UCPEventApiTest.o

UCPEventApiTest.o : UCPEventApiTest.cpp
   $(CC) -c $(UCPDV_CFLAGS) -FoUCPEventApiTest.o $(UCPDV_INC)
   UCPEventApiTest.cpp

clean :
        del UCPEventApiTest.o
        del UCPEventApiTest.exe
```

Java Approach



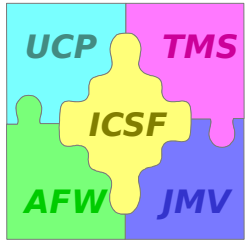
- *Support the one to one JNI wrappers as an Adapter class and implement the new model through an interface (that the Adapter class implements and a Factory class creates).*

Design Patterns



- *Adapter - "Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces."*
- *Factory - "Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses".*
- *Visitor - "Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates."*

UCP Sample Code - Java




```
import java.io.*;
import disa.ucp.UcpAdapter.*;
public class UcpChnlStatsExample {
      public static void main(String[] args) {
        try {
            IUcpContext ucpc = UcpFactory.newUcpContext();
            IUcpChannel channel;
    IUcpChnlList chnl_list = ucpc.loadChannel();
    //gets first channel
    channel = chnl_list.getFirstChnl();
            System.out.println(" Channel: " + channel.getName() + " BackLog: " +
    channel.getBackLog() + "  lastRX: " + channel.getTotString() +
                  "  LastTX: " + channel.getTorString());
            //get the rest of the channels
            while ((channel = chnl_list.getNextChnl()) != null) {
                System.out.println(" Channel: " + channel.getName() + " Backlog: " +
               channel.getBackLog() + "  lastRX: " + channel.getTotString() +
                                    "  LastTX: " + channel.getTorString());
    }
        } catch (IOException e) {
            System.err.println("Error occurred: " + e);
            System.exit(1);
        }
    }
}
```